

10-02-00

A

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT

Patent Application Transmittal

Box Patent Application
 Assistant Commissioner for Patents
 Washington, D.C. 20231

jc511 U.S. PTO
 09/675698
 09/29/00

Sir:

Transmitted herewith for filing is the Patent Application of:

Inventor: CHRISTOPHER T. GLOE, DAWN M. MAY, MARK L. BAUMAN, JAY R. WEEKS,
 PAUL R. CHMIELEWSKI

For: METHOD, APPARATUS AND ARTICLE OF MANUFACTURE FOR TRACKING
 PROCESSES

Enclosed are:

- ☒ 19 sheets of specification and 1 abstract
☒ 4 sheets of drawings
☒ A Declaration and Power of Attorney
☐ An Information Disclosure Statement and form PTO-1449
☐ A certified copy of a ___ application
☒ An assignment of the invention to International Business Machines Corporation, Armonk, New York 10504 and Recordation form PTO-1595

The filing fee has been calculated as follows:

Other Than Small Entity

For:	No. Filed	No. Extra
Basic Fee		
Total Claims	39 -20 =	19
Indep. Claims	5 -3 =	2
<input type="checkbox"/> Multiple Dependent Claim Presented		

Rate	Fee
	\$ 690.00
x \$18.00=	\$342.00
x \$78.00=	\$156.00
\$260.00	\$ 0.00
TOTAL	\$1188.00

EXPRESS MAIL CERTIFICATE

Express Mail Label No.: EL684620461USDate: SEPTEMBER 29, 2000

I hereby certify that the enclosed or attached paper is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service on the above date, addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.


 Gero G. McClellan

Patent Application Transmittal
 Attorney Docket No.: ROC920000249

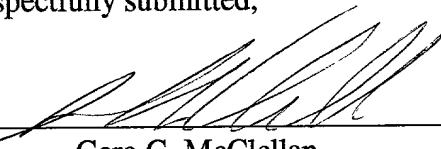
Deposit Account Authorization:

- ☒ Please charge Deposit Account No. 09-0465 in the amount of \$1188.00. A duplicate copy of this sheet is enclosed.
- ☒ The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account 09-0465. A duplicate copy of this sheet is enclosed.
- ☒ Any additional filing fees required under 37 C.F.R. §1.16.
- ☒ Any patent application processing fees under 37 C.F.R. §1.17.

Respectfully submitted,

Date: 9-29-00

By


Gero G. McClellan
Registration No.: 44,227

IBM Corporation
Intellectual Property Law, Dept. 917
3605 Highway 52 North
Rochester, MN 55901-7829

(507) 253-4660 voice
(507) 253-2382 fax

Patent Application Transmittal
Attorney Docket No.: ROC920000249

UNITED STATES PATENT APPLICATION FOR:

**METHOD, APPARATUS AND ARTICLE OF MANUFACTURE
FOR TRACKING PROCESSES**

INVENTORS:

CHRISTOPHER T. GLOE

DAWN M. MAY

MARK L. BAUMAN

JAY R. WEEKS

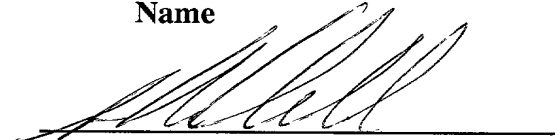
PAUL R. CHMIELEWSKI

Certification Under 37 CFR 1.10

I hereby certify that this New Application and the documents referred to as enclosed therein are being deposited with the United States Postal Service on September 29, 2000, in an envelope marked as "Express Mail United States Postal Service," Mailing Label No. EL684620461US to: Assistant Commissioner for Patents, Box PATENT APPLICATION, Washington, D.C. 20231.

Gero G. McClellan

Name



Signature

9-29-00

Date of Signature

METHOD, APPARATUS AND ARTICLE OF MANUFACTURE FOR TRACKING PROCESSES

5 BACKGROUND OF THE INVENTION

Field of the Invention

The invention relates to providing information associated with computers. More particularly, the invention relates to a method, apparatus and article of manufacture for tracking processes.

Background of the Related Art

10 In many computer systems, particularly computer systems using multi-process or multi-threading operating systems such as the IBM AIX and the IBM OS/400, the operation of running or executing of an application program is divided into work units, commonly known as processes, jobs, tasks or threads. To perform the work of the application program, the processes use socket objects to communicate with other processes.

15 These processes may run on the same computer system or on different computer systems. For example, a server computer system may service the requests of many client computer systems. To service the request from a particular client computer, a process is created in the server computer. This process uses a socket object to communicate with a corresponding process at the client computer.

20 Information associated with the different processes and socket objects is displayed as a user interface on the display of a computer system. The current user interface programs may provide a list of active TCP/IP (Transmission Control Protocol/Internet Protocol) connections, a list of active listening local sockets or a list of work units, e.g., processes, jobs or threads. However, as computer networks include multiple server and client computer systems, a large number of processes are running on the various computer systems. Thus, there is a need to improve tracking of related
25 processes, e.g., those processes using a common socket object for communication.
30

SUMMARY OF THE INVENTION

The invention provides a method, apparatus and computer readable medium for

tracking processes utilized to execute an application program. In one embodiment, the method creates a process list for a socket object, such that the process list contains a process identifier for a first process using the socket object. If a second process is using the socket object, the method updates the process list to include the process identifier for the second process.

In another embodiment, the method displays a user interface containing a list of socket objects used for a computer system, and provides a process list for a socket object selected by a user from the list of socket objects. The process list contains a process identifier for the at least one process using the selected socket object.

An apparatus comprising a memory, a processor and a display device is also provided. The memory stores a user interface program. The microprocessor executes the user interface program retrieved from the memory to create and update the process list for the socket object. The process list contains a process identifier for a first process using the socket object. If a second process is using the socket object, the process list is updated to include the process identifier of the second process.

Additionally, a computer readable medium storing a software program is provided, when the software program, when executed by a processor of a computer, causes the computer to execute a method. In one embodiment of the computer readable medium, the method creates a process list for the socket object, such that the process list contains a process identifier for a first process using the socket object. If a second process is using the socket object, the method updates the process list to include the process identifier for the second process. In another embodiment of the computer readable medium, the method displays a user interface containing a list of socket objects used for a computer system, and provides a process list for a socket object selected by a user from the list of socket objects. The process list contains a process identifier for the at least one process identifier using the selected socket object.

BRIEF DESCRIPTION OF THE DRAWINGS

The teachings of the present invention can be readily understood by considering the following detailed description in conjunction with the accompanying drawings, in which:

FIG. 1 depicts a block diagram of a computer system utilized to implement the present invention;

FIG. 2A depicts an exemplary user interface utilized to display a process list of the present invention;

FIG. 2B depicts the process list for a socket object selected from the user interface of FIG. 2A;

FIG. 3 depicts a software process diagram for creating and updating the process list of the present invention; and

FIG. 4 depicts a flow diagram of a method for implementing the process list of the present invention

To facilitate understanding, identical reference numerals have been used, where possible, to designate identical elements that are common to the figures.

It is to be noted, that the appended drawings illustrate only typical embodiments of this invention and are therefore not to be considered limiting of its scope, for the invention may admit to other equally effective embodiments.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Embodiments of the invention provide a method, apparatus and computer readable medium for tracking processes utilized to execute an application program. Initially, a process list for a socket object is created, such that the process list contains a process identifier for a first process using the socket object. If a second process is using the socket object, the process list is updated to include the process identifier for the second process.

The invention is embodied in a sockets support program utilized to create and update the process list for each socket object. Once the process list is created for a particular socket object, a user or network administrator may use a user interface program, e.g., a NETSTAT (Network Status) program, to view the process list. By viewing the process list, the user may determine whether a problem, e.g., a communications problem, exists in a process using a particular socket object coupled to another computer.

Although the following description is described in a context of tracking processes, the present invention also applies to other units of work utilized to execute an application program. For example, the present invention may similarly apply to

other commonly known units of work such as jobs, tasks, threads, and the like.

FIG. 1 depicts a computer system 100 illustratively utilized in accordance with the invention. The computer system 100 may represent any type of computer, computer system or other programmable electronic device, including a client
5 computer, a server computer, a portable computer, an embedded controller, and the like. The computer system 100 may be a standalone device or coupled to a computer network system. In one embodiment, the computer system 100 is an AS/400 available from International Business Machines of Armonk, New York.

The computer system 100 is shown in a multi-user programming environment
10 having at least one processor 102, which obtains instructions and from a memory 106 via a bus 104. The main memory 106 includes an operating system 108, various application programs including a sockets support program 110 and a user interface program 111, and various data structures including a process list 112 of the present invention. The main memory 106 may comprise one or a combination of memory
15 devices, including Random Access Memory, nonvolatile or backup memory, (e.g., programmable or flash memories, read-only memories, and the like). In addition, memory 106 may include memory physically located elsewhere in a computer system, for example, any storage capacity used as virtual memory or stored on a mass storage device, or on another computer coupled to the computer system 100.

The operating system 108 is the software utilized to operate the computer
20 system 100. More specifically, the operating system 108 performs a variety of functions including dealing with computer hardware, providing a user interface, performs user commands or program instructions and coordinates the various interfaces of the computer system 100. Additionally, the operating system 108
25 supports various functions in accordance with a Sockets Application Program Interface (API). Examples of such Sockets API based operating system 108 include UNIX, IBM OS/400, IBM AIX and Microsoft Windows.

To perform the various functions for the computer system 100, the operating
30 system 108 executes a variety of applications or application programs (not shown). The operating system 108 partitions the function or operation of each application into one or more work units, commonly known as processes, jobs, tasks or threads. Each process or job comprises one or more threads. These processes may be provided from the same computer system 100 or different computer systems, e.g., server and client

computers. For example, a process in a computer system may use a socket object 114 (also known as sockets) to communicate with another process in a different computer system. These computer systems may operate on different operating system platforms.

5 The computer system 100 is generally coupled to a number of peripheral devices. Illustratively, the computer system 100 is coupled to a direct access storage device (DASD) 122, input devices 124, output devices 126, and a plurality of networked devices 128. Each of the peripheral systems is operably coupled to the computer system 100 via respective (hardware) interfaces. A storage interface 116
10 couples the computer system 100 to the DASD 122. An input/output interface 118 couples the computer system 100 to the input device 124 and the output device 126.

In addition, a network interface 120 couples the computer system 100 to the plurality of networked devices 128. These networked devices 128 may comprise client computers if the computer system 100 comprises a server computer. The network interface 120 may comprise one or more network interface cards to connect server and client computers. In the case where the computer system 100 executes applications or application programs using the Sockets API, the network interface 120 is coupled to the processes via one or more socket objects 114. The processes may be from the same computer system 100 or from different computer systems.

15 The input device 124 may comprise any device utilized to provide input to the computer system 100. A user or a system administrator may use the input device 122 to enter a user interface command, e.g., NETSTAT (Network Status), to access the process list 112 from the memory 106. Examples of input devices 124 include a keyboard, a keypad, a light pen, a touch screen, a button, a mouse, a track ball, a speech recognition unit, and the like. The output devices 126 may comprise any
20 conventional display screen utilized to display the process list 112 retrieved from the memory 106. Although shown separately from the input devices 124, the output devices 126 and input devices 124 could be combined. For example, a display screen with an integrated touch screen, and a display with an integrated keyboard, or a speech recognition unit combined with a text speech converter could be used.

25 In general, the routines executed to implement the embodiments of the invention, whether implemented as part of an operating system 108 or a specific application, component, program, object, module or sequence of instructions will be referred to herein as a sockets support program 110, or simply as the program 110.
30

The program 110 typically comprises one or more instructions that are resident at various times in various memory and storage devices in the computer system 100. When read and executed by one or more processors 102 in the computer system 100, the program 110 causes that computer system 100 to perform the steps necessary to execute steps or elements embodying the various aspects of the invention. Moreover, while the invention has and hereinafter will be described in the context of fully functioning computers and computer systems, those skilled in the art will appreciate that the various embodiments of the invention are capable of being distributed as a program product in a variety of forms, and that the invention applies equally regardless of the particular type of signal bearing media used to actually carry out the distribution. Examples of signal bearing media include, but are not limited to, recordable type media such as volatile and nonvolatile memory devices, floppy and other removable disks, hard disk drives, optical disks (e.g., CD-ROM, DVD, and the like), among others, and transmission type media such as digital and analog communication links.

In addition, various programs and devices described hereinafter may be identified based upon the application for which they are implemented in a specific embodiment of the invention. However, it should be appreciated that any particular program or device nomenclature that follows is used merely for convenience, and the invention is not limited to use solely in any specific application identified and/or implied by such nomenclature.

The sockets support program 110 tracks processes using a socket object 114 utilized to execute an application program. Initially, a process list for a socket object 114 is created, such that the process list 112 contains a process identifier for a first process using the socket object 114. If a second process is using the socket object 114, the process list 112 is updated to include the process identifier for the second process. The process list 112 for the socket object 114 is provided on the display device 126 upon executing a user interface program 111.

FIG. 2A depicts an exemplary user interface 200 utilized to display the process list 112 of the present invention. The user interface 200 is displayed on the output device 126. A computer system 100 provides the user interface 200 upon running the user interface program 111, e.g., NETSTAT, in response an input command via the input device 124. The user interface 200 illustratively displays a list 202 of client

computers communicating with the socket objects 114 at the server computer 100.

Each row in the list 202 of client computers comprises information for a socket object or local port at the server computer system 100. For each socket object 114, the list 202 provides the remote (client) Internet Protocol (IP) address, the remote port number of the client computer, and the local port used by the socket object 114. The list 202 also provides the state of communication between the server computer 100 and the client computers. This state of communication depends on the communications protocol between the server computer and client computers. Examples of the protocol include the Transmission Control Protocol (TCP) for connection-orientated communication and the User Datagram Protocol (UDP) for connectionless communication.

A user or network administrator may use an option, e.g., 8=Display jobs, at the user interface 200 to display the process list 112 for a particular port used by the socket object 114. FIG. 2B depicts such an exemplary process list 112 containing a list of process identifiers of active processes using a particular socket object 114 selected from the process list 202 depicted in FIG. 2A. Each row of the process list 112 contains a unique process identifier 210 for a particular process. The process identifier 210 may comprise a process name 212 or job name, a user 214, e.g., a client, associated with the process name, and a process number 216 or job number. The process identifier 210, as represented by the combination of elements 212-216, is generally unique for a particular process. In one embodiment, the process number identifier 210 comprises a unique process number 216, e.g., 07435 or 07439.

Although the illustrative process list 112 shows only two active processes, the process list 112 may be configured to show all active processes using a particular socket object 114. As such, the user or network administrator may use the user interface 200 to determine the whether a problem, e.g., a communications problem, is associated with a process for a particular client computer. For example, if the user suspects a problem with a client at IP address 192.168.5.2, the user may enter option "8" at the entry for the socket object 114.

Although FIGS. 2A-2B described the user interface 200 in terms of a server computer linked to a plurality of client computers, the present invention is not so limited. For example, the GUI 200 may also be provided for a client computer linked to a plurality of server computers, or a stand-alone client computer or a stand-alone

server computer. Additionally, the process list 112 may also be displayed on a graphic user interface (GUI).

FIG. 3 depicts a software process diagram for creating and updating the process list 112 of the present invention. Specifically, the software process diagram illustrates an endpoint object 302 and associated processes (jobs) $304_1, 304_2, \dots, 304_n$ utilized to create and update the process list 112. Although one endpoint object 302 is illustratively shown in the software process diagram, the invention applies to each active endpoint object 302 created by the operating system 108.

The endpoint object 302 illustratively comprises a socket object 114 and a transport object 306. The socket object 114 is created by the operating system 108 in response to a socket (), socketpair () or accept () function within a first process or job 304_1 . The socket () function initiates the creation of the socket object 114. The socketpair () function initiates the creation a pair of socket objects 114. The accept () function waits for an incoming connection from a client computer and ties the incoming connection to the socket object 114 for an application program.

The transport object 306 operates as an interface to the endpoint object 302 of another computer. The transport object 306 may comprise a TCP transport object to implement connection-orientated communication or a UDP transport object to implement connectionless communication. The creation of the transport object 306 depends on an address parameter in the socket () and socketpair () functions. If the address parameter or address field is AF_INET, a transport object 306 is created with socket object 114. The AF_INET address typically comprises a 32-bit or 128-bit Internet Protocol (IP) address for identifying a computer system 100 on a global network, e.g., the Internet. If the address parameter is AF_UNIX, only the socket object 114 is created in response to the socket () or socketpair () functions, i.e., no transport object 306 is required or created. The AF_UNIX address is utilized by the operating system 108 to communicate between two processes on the same computer system 100.

The socket object 114 is accessed or utilized by an application program using a Sockets Application Program Interface (Sockets API). Namely, the socket object 114 allows an application program to interface with the transport or network layers of the TCP/IP model or other network support models, e.g., AF_UNIX. A process list 112 of the present invention is provided for the socket object 114. In one embodiment, the

process list 112 contains a list of all the active processes using the socket object 114. However, the process list 112 may also be configured to contain a list of all processes using the socket object 114 during a predefined time interval.

The processes are related such that a first job 304₁ references a second job 304₂, which then references a third job (not shown), and continues up to the n-th job 304_n. The processes or jobs may run on the same computer system 100 or on different computer systems. For example, if the socket object 114 was created using a AF_UNIX address, the jobs may run on the same computer system 100. If the socket object 114 was created using a AF_INIT address, the jobs may run on different computer systems 100, i.e., a server computer and a client computer.

Initially, a socket (), socketpair () or accept () function in the first job 304₁ causes the operating system 108 to create the socket object 114. If the socket object 114 was created using an AF_INET address, the operating system 108 also creates the transport object 306. A process list 112 or job list is created for the socket object 114. As the process list 112 or job list contains the active processes using the socket object 114, the first entry in the process list 112 is the process issuing the socket (), socketpair () or accept () function utilized to create the socket object 114. A socket descriptor is also created for the socket object 114. The socket descriptor is an identifier or handle utilized for identifying the socket object 114 such that other processes may also use the socket object 114.

To enable a second job 304₂ to utilize the socket object 114, the first job 304₁ passes the socket descriptor to the second job 304₂ by issuing a spawn (), senddescriptor () or sendmsg () function. The spawn () function creates the second job 304₂ and passes the socket descriptor to the second job 304₂. The senddescriptor () function passes the access rights to the socket descriptor to the second job 304₂, which then issues a takedescriptor () function to receive the access rights to the socket descriptor. The sendmsg () function sends the socket descriptor to the second job 304₂, which issues a recvmsg () function to the socket descriptor.

Once the socket descriptor is received by the second job 304₂, the function used to receive the socket descriptor issues an input/output control command, e.g., SIOADDFD IOCTL, to the socket object 114. In response to the SIOADDFD IOCTL function, the second job 304₂ is added to the process list 112 for the socket object 114. If the socket descriptor is passed to additional processes within a sequence of n

processes or jobs 304₁, 304₂,...304_n using the socket object 114, these processes 304₁, 304₂,...304_n are also added to the process list 112. Once a process expires or no longer utilizes the socket object 114, another input/output control command, a SIOREMFDF IOCTL command is issued to the socket object 114, which causes the job to be removed from the process list 112.

In accordance to the present invention, the process list 112 for the socket object 114 is provided on the display 126 of a computer system 100. A user, e.g., a system or network administrator, may use a user interface program 110, e.g., a NETSTAT application 308, to display the process list 112 using a particular socket object.

Namely, one embodiment of the present invention may provide the process list 112 as an option in the NETSTAT application 308. For example, the network administrator may use a GetProcessList command to select an endpoint object 302 (containing the socket object 114) to view the process list 112 for the selected endpoint object 302 or socket object 114. To view all processes associated with a remote client computer, the administrator may select all the endpoint objects 302 for the particular client computer and obtain the process list 112 for each selected endpoint object 302 or socket object 114 as shown in FIG. 2B.

FIG. 4 depicts a flow diagram of a method 400 for implementing the process list 112. Although the method 400 is described in terms of a single socket object 114, the method 400 also applies to other socket objects 114. Specifically, the method 400 starts at step 402 and proceeds to step 404, where the socket object 114 is created by the operating system 108. The creation of the socket object 114 occurs in response to a process initiating the use of the socket object 114 for an application program. Namely, the process issues a socket (), socketpair () or accept () function to create the socket object 114.

At step 406, the method 400 creates and initializes a process list 112 for the socket object 114. In one embodiment, the process list 112 contains the process identifier a list of active processes using the socket object 114. Initially, the process list 112 contains the process identifier of the process containing the function utilized to create the socket object 114. The method 400 proceeds to step 408, where a socket descriptor is assigned or provided for the socket object 114. The socket descriptor operates as an identifier or handle for all processes actively using the socket object 114.

The method 400 proceeds to step 410, where a query determines whether the socket descriptor is passed to another process, e.g., a second process or job. A socket descriptor is passed to another process when multiple processes are desired for a particular socket. To receive the socket descriptor, the other process may issue a
5 takedescriptor () function in response to a senddescriptor () function issued by an active process using the socket object, or issue a recvmsg () function in response to a sendmsg () function issued by an active process using the socket object 114. The other process may also receive the socket descriptor if the process is created in response to a spawn () function issued by an active process using the socket object
10 114. In any case, upon receipt of the socket descriptor by the other process, the function used to receive the socket descriptor issues an input/output control command, e.g., SIOADDFD IOCTL, to the socket object 114. Thus, step 410 determines whether the socket object 114 has received a SIOADDFD IOCTL command in response to a function utilized to pass a socket descriptor to another process.

If the socket descriptor is passed to another process, the method 400 proceeds
15 to step 412, where the process identifier 210 for the process receiving the socket descriptor is added to the process list 112. After adding the process identifier 210 to the process list 112, the method 400 returns to step 410. If the socket descriptor is not passed to another process, the method 400 proceeds to step 414, where a query
20 determines whether the socket descriptor is removed from a process using the socket object 114. For example, when a process no longer requires the use of the socket object 114, the process may issue a close () function. Once the close () function is issued from the process, the close () function causes the removal of the socket descriptor from the process issuing the close () function, and issues an input/output control command, e.g., a SIOREMFDFD IOCTL command, to the socket object 114.
25 Thus, step 414 determines whether the socket object 114 has received a SIOREMFDFD IOCTL command in response to the close () function issued by a process.

If the socket descriptor is removed from the process, the method 400 proceeds
30 to remove the associated process identifier 210 from the process list 112 at step 416 and returns to step 410. If the socket descriptor is not removed from the process, the method 400 proceeds to step 418, where a query determines whether a process using the socket object 114 has expired. If the process has expired, the method 400 proceeds to step 420, where the process identifier 210 for the expired process is removed from

the process list 112. After removing the process identifier 210 from the process list 112, the method 400 returns to step 410. An input/output control command, e.g., SIMREMFID IOCTL, is transmitted to the socket object when the process expires. If the process has not expired, the method 400 proceeds to step 422, where a query
5 determines whether the process list 112 is empty, i.e., no longer contains any processes. If the process list is not empty, the method 400 returns to step 410. If the process list is empty, i.e., when the last process 304_n using the socket expires or issues the close () function, the method 400 proceeds to step 424, where the socket object 114 and associated process list 112 are deleted. The method 400 ends at step 426.

10 While the foregoing is directed to the preferred embodiment of the present invention, other and further embodiments of the invention may be devised without departing from the basic scope thereof, and the scope thereof is determined by the claims that follow.

WHAT IS CLAIMED IS:

1. Method for tracking at least one process using a socket object, where the at least one process is utilized to execute an application program, the method comprising: creating a process list for the socket object, where the process list contains a process identifier for a first process using the socket object; and updating, if a second process is using the socket object, the process list to include the process identifier for the second process.

2. The method of claim 1 wherein the process list is displayed on a user interface in response to a user interface command entered by a user.

3. The method of claim 1 wherein the step of updating comprises: adding the process identifier of the second process to the process list if the second process is to use the socket object.

4. The method of claim 3 wherein the second process is to use the socket object if a socket descriptor created for the socket object is passed from the first process to the second process.

5. The method of claim 1 wherein the step of updating comprises: removing the process identifier of at least one of the first process and second process from the process list if the at least one of the first process and second process no longer uses the socket object.

6. The method of claim 5 wherein the at least one of the first process and second process no longer uses the socket object if a socket descriptor created for the socket object is removed from the at least one of the first process and second process.

7. The method of claim 1 wherein the step of updating comprises: removing the process identifier of at least one of the first process and the second process from the process list if the at least one of the first process and second process expires.

- 1 8. The method of claim 1 wherein the first process comprises a Sockets
2 Application Program Interface (API) function utilized to create the socket object.
3
- 4 9. The method of claim 8 wherein the Sockets API function comprises one of a
5 socket () function, a socketpair () function and an accept () function.
6
- 7 10. The method of claim 1 wherein the creating and updating are performed by an
8 operating system after a computer executes a sockets support program.
9
- 10 11. The method of claim 1 wherein the first process and the second process are
11 provided in the same computer system.
12
- 13 12. The method of claim 1 wherein the first process and the second process are
14 provided in different computer systems.
15
- 16 13. The method of claim 1 wherein the process identifier comprises at least one of a
17 process name, a user name associated with the process name and a process number.
18
- 19 14. A method for administering network information, the method comprising:
20 displaying a user interface containing a list of socket objects used for a
21 computer system; and
22 providing a process list for a socket object selected by a user from the list of
23 socket objects, where the process list contains a process identifier for at least one
24 process using the selected socket object..
25
- 26 15. The method of claim 14 wherein each of the at least one process using the
27 selected socket object comprises an active process using the socket object.
28
- 29 16. The method of claim 14 wherein each socket object in the list of socket objects
30 is associated with a client computer communicating with a server computer.
31
- 32 17. The method of claim 14 wherein the process identifier comprises at least one of
33 a process name, a user name associated with the process name and a process number.

1
2 18. An apparatus for tracking at least one process of an application program using
3 a socket object, the apparatus comprising:

4 a memory for storing an operating system and a sockets support program; and
5 a processor, coupled to the memory, for performing a method upon executing
6 the sockets support program retrieved from the memory, the method comprising:

7 creating the process list for a socket object, where the process list
8 contains a process identifier for a first process using the socket object, and
9 updating, if a second process is using the socket object, the process list
10 to include the process identifier of the second process.

11
12 19. The apparatus of claim 18 further comprising:

13 a network interface for coupling the socket object with a remote device.
14

15 20. The apparatus of claim 18 further comprising:

16 a display device, coupled to the processor, for displaying the process list when
17 the processor retrieves and executes a user interface program from the memory.
18

19 21. The apparatus of claim 18 wherein the operating system comprises one of
20 UNIX, IBM AIX, IBM OS/400 and Microsoft Windows.
21

22 22. The apparatus of claim 18 wherein the network interface couples the first
23 process to the second process.
24

25 23. A computer readable medium storing a software program that, when executed
26 by a processor of a computer, causes the computer to perform a method comprising:

27 creating a process list for a socket object, where the process list contains a
28 process identifier for a first process using the socket object; and
29 updating, if the second process is using the socket object, the process list to
30 include the process identifier of the second process.
31

24. The computer readable medium of claim 23 wherein the process list is displayed on a user interface in response to a user interface command entered by a user.

25. The computer readable medium of claim 23 wherein the step of updating comprises:

adding the process identifier of the second process to the process list if the second process is to use the socket object.

26. The computer readable medium of claim 25 wherein the second process is to use the socket object if a socket descriptor created for the socket object is passed from the first process to the second process.

27. The computer readable medium of claim 23 wherein the step of updating comprises:

removing the process identifier of at least one of the first process and second process from the process list if the at least one of the first process and second process no longer uses the socket object.

28. The computer readable medium of claim 27 wherein the at least one of the first process and second process no longer uses the socket object if a socket descriptor created for the socket object is removed from the at least one of the first process and second process.

29. The computer readable medium of claim 23 wherein the step of updating comprises:

removing the process identifier of at least one of the first process and the second process from the process list if the at least one of the first process and second process expires.

30. The computer readable medium of claim 23 wherein the first process comprises a Sockets Application Program Interface (API) function utilized to create the socket object.

1
2 31. The computer readable medium of claim 30 wherein the Sockets API function
3 comprises one of a socket () function, a socketpair () function and an accept ()
4 function.

5
6 32. The computer readable medium of claim 23 wherein the creating and updating
7 are performed by an operating system after a computer executes a sockets support
8 program.

9
10 33. The computer readable medium of claim 23 wherein the first process and the
11 second process are provided in the same computer system.

12
13 34. The computer readable medium of claim 23 wherein the first process and the
14 second process are provided in different computer systems.

15
16 35. The computer readable medium of claim 23 wherein the process identifier
17 comprises at least one of a process name, a user name associated with the process
18 name and a process number.

19
20 36. A computer readable medium storing a software program that, when executed
21 by a processor of a computer, causes the computer to perform a method comprising::
22 displaying a user interface containing a list of socket objects used for a
23 computer system; and
24 providing a process list for a socket object selected by a user from the list of
25 socket objects, where the process list contains a process identifier for at least one
26 process using the selected socket object.

27
28 37. The computer readable medium of claim 36 wherein each of the at least one
29 process using the selected socket object comprises an active process using the socket
30 object.

39. The computer readable medium of claim 36 wherein the process identifier comprises at least one of a process name, a user name associated with the process name and a process number

a) $\Delta_{\text{max}} = 0.001$	
Δ_{max}	$\Delta_{\text{max}} = 0.001$
0.001	0.001
0.002	0.002
0.003	0.003
0.004	0.004
0.005	0.005
0.006	0.006
0.007	0.007
0.008	0.008
0.009	0.009
0.010	0.010
0.011	0.011
0.012	0.012
0.013	0.013
0.014	0.014
0.015	0.015
0.016	0.016
0.017	0.017
0.018	0.018
0.019	0.019
0.020	0.020
0.021	0.021
0.022	0.022
0.023	0.023
0.024	0.024
0.025	0.025
0.026	0.026
0.027	0.027
0.028	0.028
0.029	0.029
0.030	0.030
0.031	0.031
0.032	0.032
0.033	0.033
0.034	0.034
0.035	0.035
0.036	0.036
0.037	0.037
0.038	0.038
0.039	0.039
0.040	0.040
0.041	0.041
0.042	0.042
0.043	0.043
0.044	0.044
0.045	0.045
0.046	0.046
0.047	0.047
0.048	0.048
0.049	0.049
0.050	0.050
0.051	0.051
0.052	0.052
0.053	0.053
0.054	0.054
0.055	0.055
0.056	0.056
0.057	0.057
0.058	0.058
0.059	0.059
0.060	0.060
0.061	0.061
0.062	0.062
0.063	0.063
0.064	0.064
0.065	0.065
0.066	0.066
0.067	0.067
0.068	0.068
0.069	0.069
0.070	0.070
0.071	0.071
0.072	0.072
0.073	0.073
0.074	0.074
0.075	0.075
0.076	0.076
0.077	0.077
0.078	0.078
0.079	0.079
0.080	0.080
0.081	0.081
0.082	0.082
0.083	0.083
0.084	0.084
0.085	0.085
0.086	0.086
0.087	0.087
0.088	0.088
0.089	0.089
0.090	0.090
0.091	0.091
0.092	0.092
0.093	0.093
0.094	0.094
0.095	0.095
0.096	0.096
0.097	0.097
0.098	0.098
0.099	0.099
0.100	0.100
0.101	0.101
0.102	0.102
0.103	0.103
0.104	0.104
0.105	0.105
0.106	0.106
0.107	0.107
0.108	0.108
0.109	0.109
0.110	0.110
0.111	0.111
0.112	0.112
0.113	0.113
0.114	0.114
0.115	0.115
0.116	0.116
0.117	0.117
0.118	0.118
0.119	0.119
0.120	0.120
0.121	0.121
0.122	0.122
0.123	0.123
0.124	0.124
0.125	0.125
0.126	0.126
0.127	0.127
0.128	0.128
0.129	0.129
0.130	0.130
0.131	0.131
0.132	0.132
0.133	0.133
0.134	0.134
0.135	0.135
0.136	0.136
0.137	0.137
0.138	0.138

ABSTRACT OF THE DISCLOSURE

5 A method, apparatus and computer readable medium is provided for tracking processes using a socket object. The processes are utilized to execute an application program. Initially, a process list for the socket object is created, such that the process list contains a process identifier for a first process using the socket object. If a second process is using the socket object, the process list is updated to include the process identifier for the second process.

┌

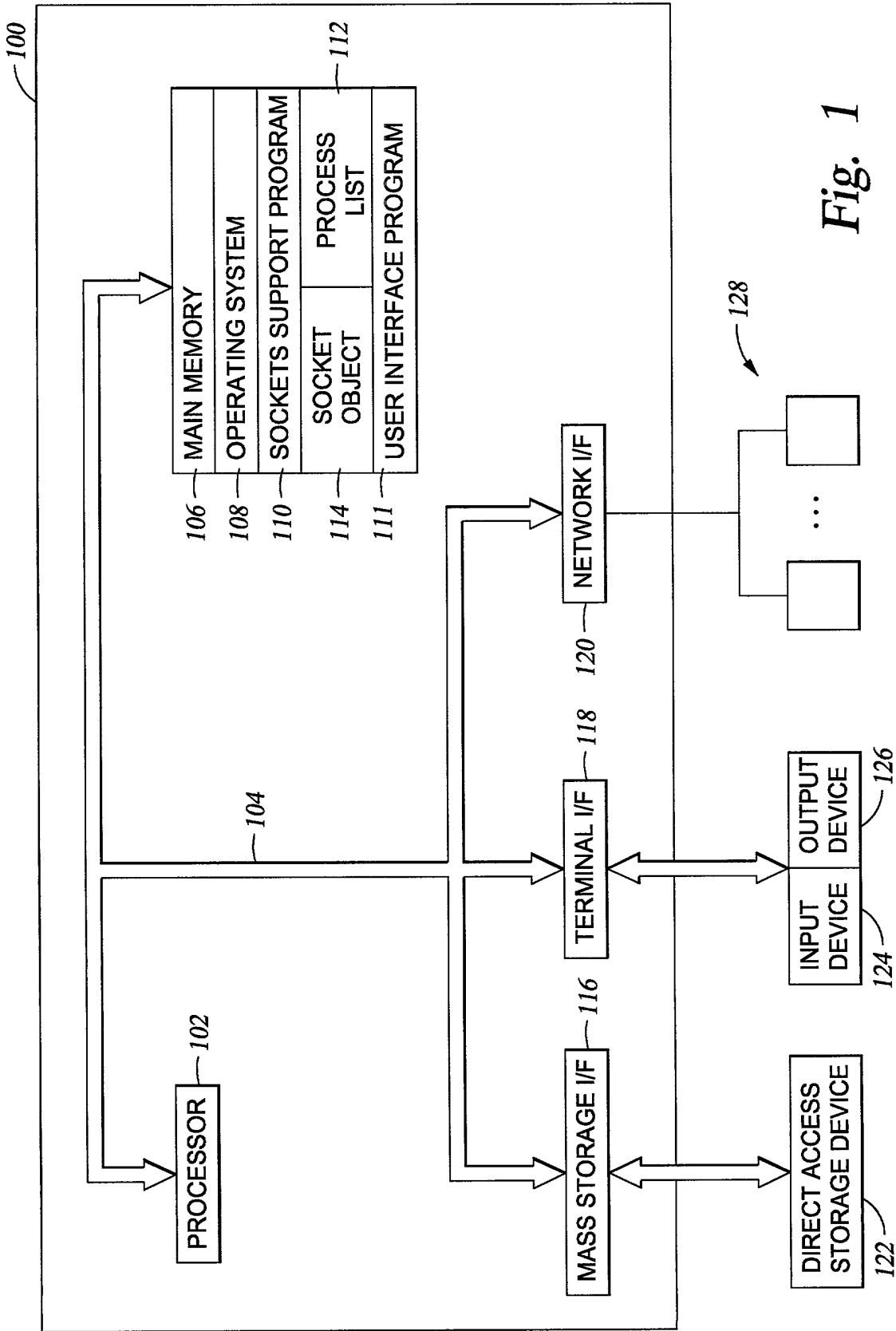


Fig. 1

└

WORK WITH TCP/IP CONNECTION STATUS

SYSTEM RS043

TYPE OPTIONS, PRESS ENTER.
 3 = ENABLE DEBUG 4 = END 5 = DISPLAY DETAILS 6 = DISABLE DEBUG
 8 = DISPLAY JOBS

OPT	REMOTE ADDRESS	REMOTE PORT	LOCAL PORT	IDLE TIME	STATE
—	9.130.69.80	34266	TELNET	000:16:36	ESTABLISHED
—	9.130.69.108	3032	TELNET	000:03:05	ESTABLISHED
—	9.130.69.236	1069	*	005:07:15	*UDP
—	9.130.69.236	1073	*	005:07:15	*UDP
—	192.168.5.2	1031	*	005:07:15	*UDP
8	192.168.5.2	1031	5033	000:29:37	ESTABLISHED

F5 = REFRESH F11= DISPLAY BYTE COUNTS F13 = SORT BY COLUMN
 F14 = DISPLAY PORT NUMBERS F22 = DISPLAY ENTIRE FIELD
 F24 = MORE KEYS

Fig. 2A

WORK WITH TCP/IP CONNECTION STATUS

SYSTEM RS043

CONNECTION TYPE *TCP
 LOCAL ADDRESS 192.168.5.1
 LOCAL PORT 5033
 REMOTE ADDRESS 192.168.5.2
 REMOTE PORT 1031

TYPE OPTIONS, PRESS ENTER.
 5= WORK WITH JOB

OPT	NAME	USER	NUMBER
—	RS043NWS	QSYS	07435
—	RS043NWS	QSYS	07439

F3 = EXIT F5 = REFRESH F6 = PRINT F9 = COMMAND LINE
 F12 = CANCEL

Fig. 2B

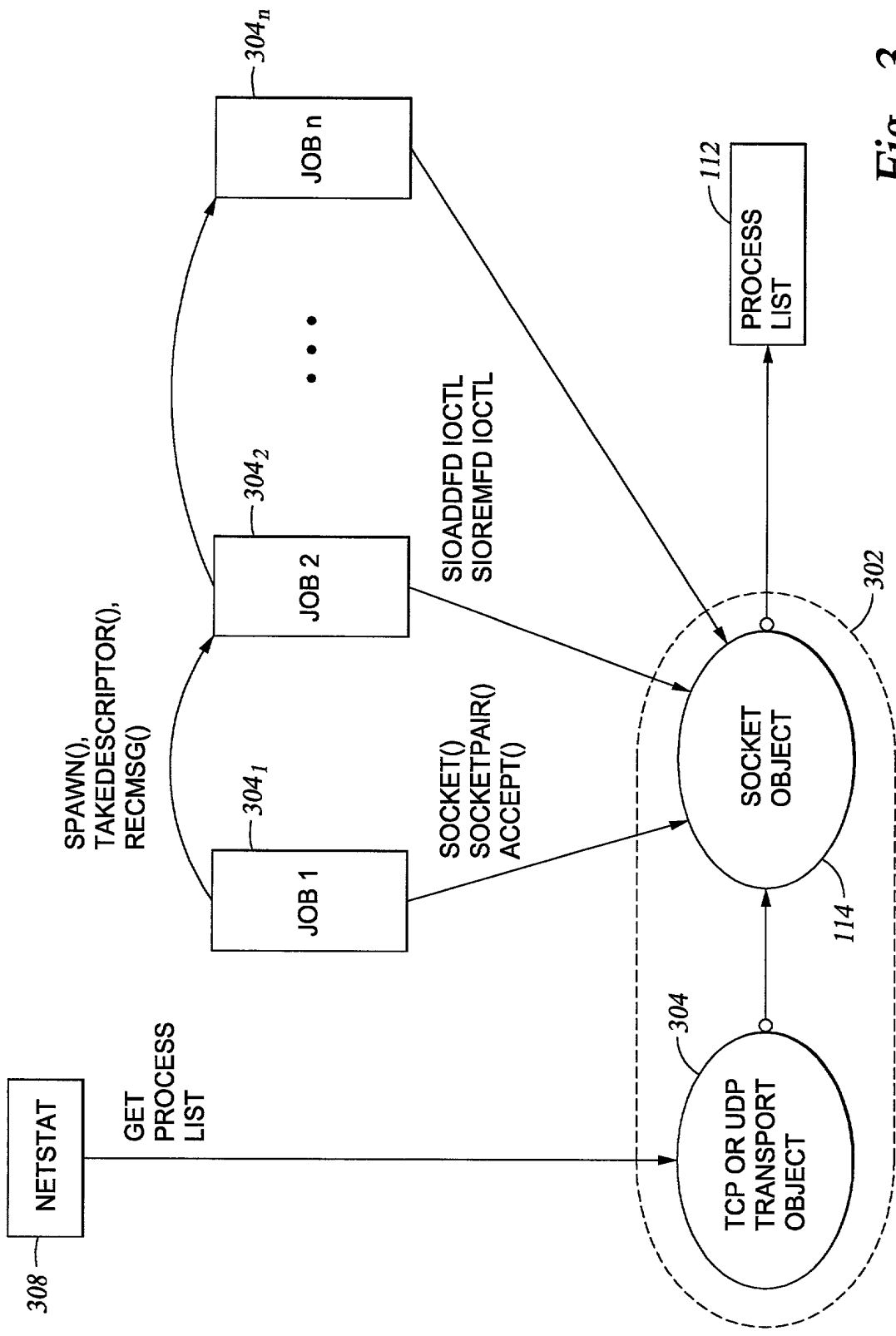


Fig. 3

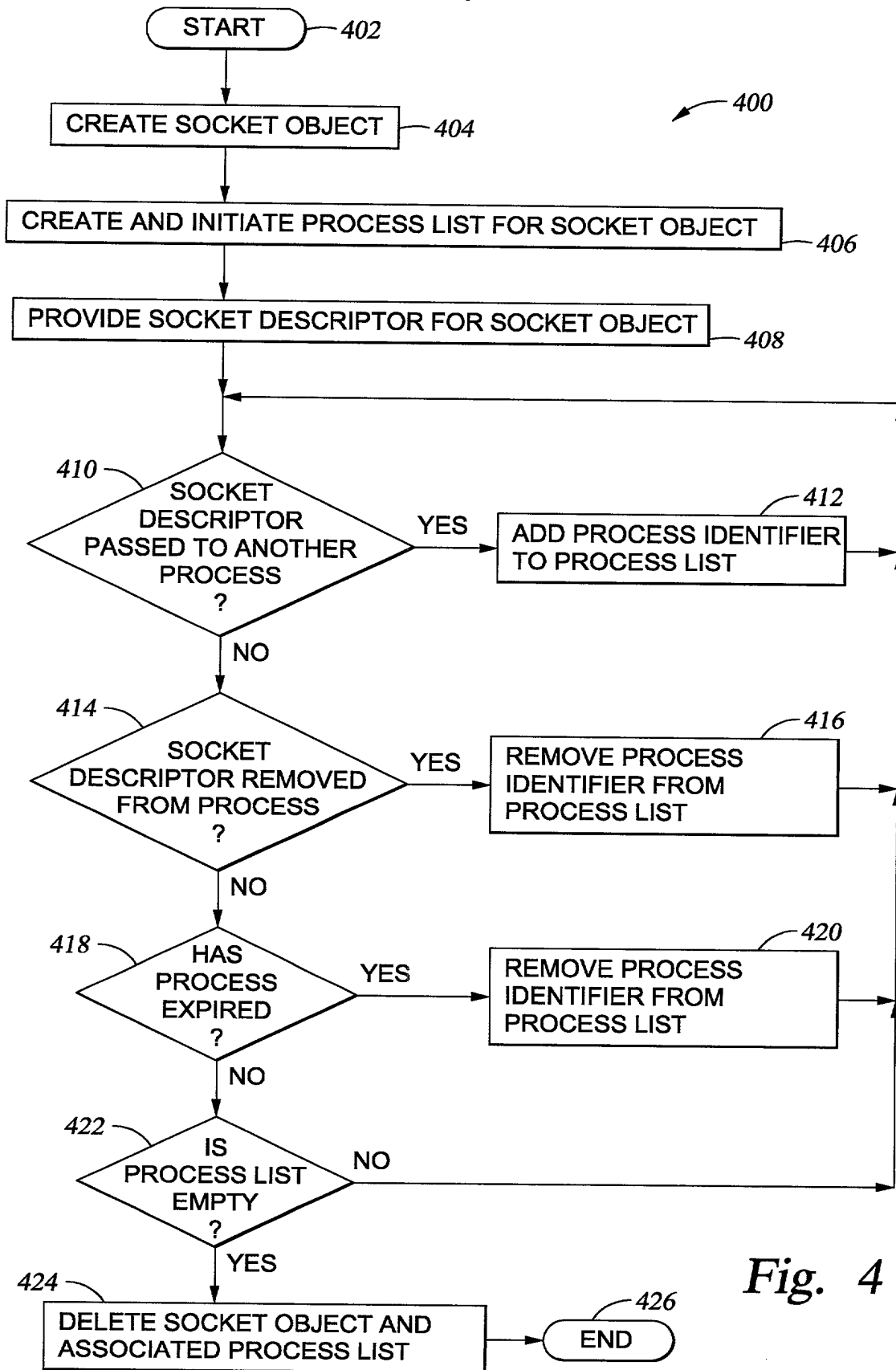


Fig. 4

Table 1. Demographic characteristics of the study population	
Age (years)	Mean (SD)
18-24	20.5 (2.5)
25-34	29.5 (4.5)
35-44	39.5 (5.5)
45-54	49.5 (6.5)
55-64	59.5 (7.5)
65-74	69.5 (8.5)
75-84	79.5 (9.5)
85-94	89.5 (10.5)
95-104	99.5 (11.5)
105-114	109.5 (12.5)
115-124	119.5 (13.5)
125-134	129.5 (14.5)
135-144	139.5 (15.5)
145-154	149.5 (16.5)
155-164	159.5 (17.5)
165-174	169.5 (18.5)
175-184	179.5 (19.5)
185-194	189.5 (20.5)
195-204	199.5 (21.5)
205-214	209.5 (22.5)
215-224	219.5 (23.5)
225-234	229.5 (24.5)
235-244	239.5 (25.5)
245-254	249.5 (26.5)
255-264	259.5 (27.5)
265-274	269.5 (28.5)
275-284	279.5 (29.5)
285-294	289.5 (30.5)
295-304	299.5 (31.5)
305-314	309.5 (32.5)
315-324	319.5 (33.5)
325-334	329.5 (34.5)
335-344	339.5 (35.5)
345-354	349.5 (36.5)
355-364	359.5 (37.5)
365-374	369.5 (38.5)
375-384	379.5 (39.5)
385-394	389.5 (40.5)
395-404	399.5 (41.5)
405-414	409.5 (42.5)
415-424	419.5 (43.5)
425-434	429.5 (44.5)
435-444	439.5 (45.5)
445-454	449.5 (46.5)
455-464	459.5 (47.5)
465-474	469.5 (48.5)
475-484	479.5 (49.5)
485-494	489.5 (50.5)
495-504	499.5 (51.5)
505-514	509.5 (52.5)
515-524	519.5 (53.5)
525-534	529.5 (54.5)
535-544	539.5 (55.5)
545-554	549.5 (56.5)
555-564	559.5 (57.5)
565-574	569.5 (58.5)
575-584	579.5 (59.5)
585-594	589.5 (60.5)
595-604	599.5 (61.5)
605-614	609.5 (62.5)
615-624	619.5 (63.5)
625-634	629.5 (64.5)
635-644	639.5 (65.5)
645-654	649.5 (66.5)
655-664	659.5 (67.5)
665-674	669.5 (68.5)
675-684	679.5 (69.5)
685-694	689.5 (70.5)
695-704	699.5 (71.5)
705-714	709.5 (72.5)
715-724	719.5 (73.5)
725-734	729.5 (74.5)
735-744	739.5 (75.5)
745-754	749.5 (76.5)
755-764	759.5 (77.5)
765-774	769.5 (78.5)
775-784	779.5 (79.5)
785-794	789.5 (80.5)
795-804	799.5 (81.5)
805-814	809.5 (82.5)
815-824	819.5 (83.5)
825-834	829.5 (84.5)
835-844	839.5 (85.5)
845-854	849.5 (86.5)
855-864	859.5 (87.5)
865-874	869.5 (88.5)
875-884	879.5 (89.5)
885-894	889.5 (90.5)
895-904	899.5 (91.5)
905-914	909.5 (92.5)
915-924	919.5 (93.5)
925-934	929.5 (94.5)
935-944	939.5 (95.5)
945-954	949.5 (96.5)
955-964	959.5 (97.5)
965-974	969.5 (98.5)
975-984	979.5 (99.5)
985-994	989.5 (100.5)
995-1004	999.5 (101.5)
1005-1014	1009.5 (102.5)
1015-1024	1019.5 (103.5)
1025-1034	1029.5 (104.5)
1035-1044	1039.5 (105.5)
1045-1054	1049.5 (106.5)
1055-1064	1059.5 (107.5)
1065-1074	1069.5 (108.5)
1075-1084	1079.5 (109.5)
1085-1094	1089.5 (110.5)
1095-1104	1099.5 (111.5)
1105-1114	1109.5 (112.5)
1115-1124	1119.5 (113.5)
1125-1134	

My residence, post office address and citizenship are as stated below next to my name. I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

the specification of which (check one)

_____ was filed on _____ as _____

and was amended on _____

(if applicable)

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, §1.56.

Prior Foreign Application(s)	Priority Claimed
------------------------------	------------------

I hereby claim the benefit under Title 35, United States Code, §120 of any United States Application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose information material to the patentability of this application as defined in Title 37, Code of Federal Regulations, §1.56(a) which occurred between the filing date of the

prior application and the national or PCT international filing date of this application:

(NONE)
(Application Serial No.) (Filing Date) (Status) (Patented, Pending, Abandoned)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

POWER OF ATTORNEY: As a named inventor I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

(List name and registration number)

William J. McGinnis - 25,698	Roy W. Truelson - 34,265
Christopher J. Hughes - 26,914	John E. Hoel - 26,279
James R. Nock - 42,937	Edward A. Pennington - 32,588
Steven W. Roth - 34,712	Joseph C. Redmond, Jr. - 18,753
Gero G. McClellan - 44,227	

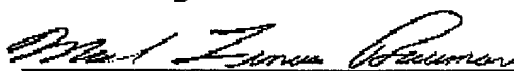
Send Correspondence to: Gero G. McClellan
Thomason, Moser & Patterson, L.L.P.
3040 Post Oak Boulevard, Suite 1500
Houston, TX 77056-6582

Direct Telephone Calls to: Gero G. McClellan
Area Code 713-623-4844

Full name of sole or first Inventor

Mark Linus Bauman
Inventor's signature

Date



9/29/2000

Residence

6115 Fairway Drive N.W., Rochester, Minnesota 55901-8700

Citizenship

U.S.A.

Post Office Address

Same as above

Full name of second Inventor

Paul Richard Chmielewski

Inventor's signature

Date _____

Paul Richard Chamberlayne

9/28/00

Residence

6005 10th Street S.W., Rochester, Minnesota 55902

Citizenship

U.S.A.

Post Office Address

Same as above

Full name of third Inventor

Christopher Thomas Gloe

Inventor's signature

Date _____

Christopher Thomas Goe

9/28/00

Residence

2191 68th Street N.W., Rochester, Minnesota 55901

Citizenship

U.S.A.

Post Office Address

Same as above

Full name of fourth Inventor

Dawn Marie May

Inventor's signature

Date _____

Dawn Marie May
Residence

9-28-00

Residence

308 Deer Ridge Court, Mantorville, Minnesota 55955

Citizenship

U.S.A.

Post Office Address

Same as above

Full name of fifth Inventor

Jay Robert Weeks

Inventor's signature

Date _____

Residence

108 4th Street, Cambridge, Iowa 50046

Citizenship

U.S.A.

Post Office Address

Same as above

X This declaration ends with this page.

[illegible]

My residence, post office address and citizenship are as stated below next to my name. I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

the specification of which (check one)

_____ was filed on _____ 25

and was amended on _____

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Priority Claimed

I hereby claim the benefit under Title 35, United States Code, §120 of any United States Application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112. I acknowledge the duty to disclose information material to the patentability of this application as defined in Title 37, Code of Federal Regulations, §1.56(a) which occurred between the filing date of the

__ (NONE)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

(List name and registration number)

William J. McGinnis - 25,698	Roy W. Truelson - 34,265
Christopher J. Hughes - 26,914	John E. Hoel - 26,279
James R. Nock - 42,937	Edward A. Pennington - 32,588
Steven W. Roth - 34,712	Joseph C. Redmond, Jr. - 18,753
Gero G. McClellan - 44,227	

Send Correspondence to: Gero G. McClellan
Thomason, Moser & Patterson, L.L.P.
3040 Post Oak Boulevard, Suite 1500
Houston, TX 77056-6582

Direct Telephone Calls to: Gero G. McClellan
Area Code 713-623-4844

Full name of sole or first Inventor

Mark Linus Bauman

Inventor's signature

Date _____

Residence

6115 Fairway Drive N.W., Rochester, Minnesota 55901-8700

Citizenship

U.S.A.

Post Office Address

Same as above

Full name of second inventor

Paul Richard Chmielewski

Inventor's signature

Date _____

Residence

6005 10th Street S.W., Rochester, Minnesota 55902

Citizenship

U.S.A.

Post Office Address

Same as above

Full name of third Inventor

Christopher Thomas Gloe

Inventor's signature

Date _____

Residence

2191 68th Street N.W., Rochester, Minnesota 55901

Citizenship

U.S.A.

Post Office Address

Same as above

Full name of fourth Inventor

Dawn Marie May

Inventor's signature

Date _____

Residence

308 Deer Ridge Court, Mantorville, Minnesota 55955

Citizenship

U.S.A.

Post Office Address

Same as above

Full name of fifth Inventor

Jay Robert Weeks

Inventor's signature

Date _____

Director's signature
Jay Robert Weeks

9/29/00

Residence

108 4th Street, Cambridge, Iowa 50046

Citizenship

U.S.A.

Post Office Address

Same as above

X This declaration ends with this page.